*Embedded Systems: Part 4*

# Nothing fuzzy about processor choice

By Joe Altnether
Fuzzy Logic Program Director
Intel Corp.
Chandler, Ariz.

However arcane fuzzy logic might seem to designers just making its acquaintance, one aspect of fuzzy design remains relatively straightforward: Choosing a microcontroller for a fuzzy system is no more wrenching than choosing one for conventional systems. If care is taken in the selection process, microcontrollers executing fuzzy software can easily meet the performance requirements of fuzzy applications.

Peripheral functionality has not changed for fuzzy logic, but new architectural requirements involve register depth and logic functionality. The number of bits is determined by the resolution required.

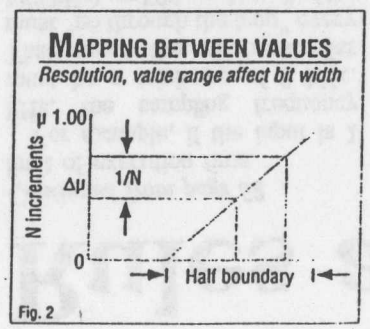

Fig. 2

The use of fuzzy logic opens the system performance envelope in two dimensions: machine IQ (application intelligence) and architecture performance. The machine IQ is a function of the microprocessor/controller performance. Conversion from traditional logic to fuzzy logic permits the designer either to retain the same processor performance and increase the machine IQ, or keep the same machine IQ and reduce the processor performance requirements (see Fig. 1).

This stretched performance is the result of using rules rather than algorithms to control the system. Where many traditional applications incorporate high-order-differential equations to solve control problems, fuzzy logic instead uses a series of rules, significantly reducing performance requirements.

This excess performance can either be eliminated to reduce the system cost or used to create a smarter system. Therefore, some applications can reduce their architectural requirements by moving from a 32-bit processor to 16 bits, or from 16 bits to 8.

Given this freedom of choice and the unfamiliarity of performance expectations of fuzzy logic, the microcontroller possibilities are bewildering. Fortunately, system requirements set the selection criteria, just as they do with conventional systems. These criteria mirror the fuzzy engine and the interface requirements.

Every control application has two elements: an interface to the "real world" for input and output, and the control strategy. The sum of these two elements defines the performance requirements. In both conventional and fuzzy logic systems, the interface is performed by the microcontroller's peripherals. Fuzzy logic implements the control strategy.

Defining the rules by which a fuzzy logic application works is a three-step process: "fuzzification," inference and "defuzzification." Fuzzification maps crisp values to a fuzzy membership, while inferencing evaluates the validity of a term, or linguistic variable, in the rule. Finally, defuzzification translates the fuzzy output to a crisp output. This crisp output enters the real world via the peripherals.

The resolution of the data to be fuzzified and its membership define the bit requirement of the microcontroller. Fuzzy logic is a mapping between crisp values, such as 78x, and fuzzy values, such as the linguistic variable "warm." This mapping transfers crisp values into the membership set or function. The membership value ranges from 0 to 1, with all increments in between valid. Most applications can tolerate a membership resolution of .01. As a result, the membership set can be represented by a byte or 8 bits.

Next, the slope of a term such as "warm" exerts its influence on the resolution. The slope shown in Fig. 2 is 1/2 × Boundary, where Boundary indicates the range of values (such as 65° to 75°F) within which the term applies. The resolution of the crisp value is the Boundary value divided by twice the membership resolution. Application resolution is the range of the universe of discourse (the total range of values for all terms) times the Boundary divided by twice the resolution of the membership.

The range of the crisp values in the universe of discourse and the range of the terms affect the bit requirements. A wide range of input values and a broad term will require more resolution, which results in a wider word requirement. In data control applications, the size of the data file to be evaluated also affects the bit-width requirements. Systems dealing with data analysis and pattern recognition have large data sets and require a wide bit width for addressability.

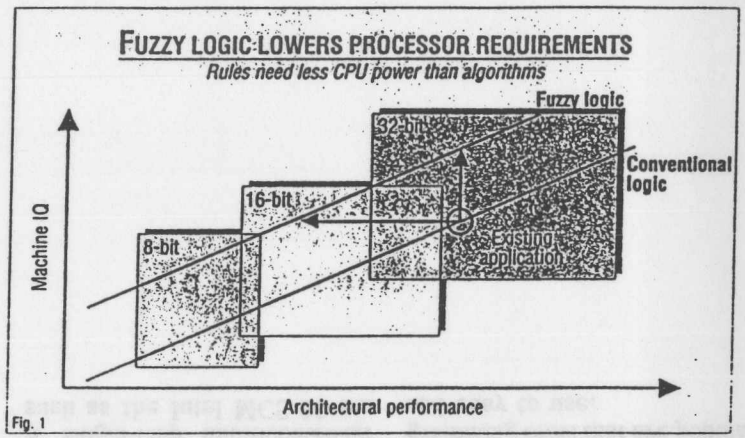

Fig. 1

## Inference rules

The set of rules replaces the control algorithm or equation used in conventional systems. Each rule in the set defines an action to be taken if evaluation conditions are met. To adequately define the system, a number of rules must be evaluated each time new input data is acquired or a new output must be generated.

Microcontrollers evaluate the rules in a serial fashion. As a consequence, increasing the number of rules increases the time needed to evaluate them. Fuzzy systems are nondeterministic, and not every rule will participate each time. Worst-case timing is calculated by assuming that every rule does participate each time.

Each time a new input sample is captured, it must be evaluated to generate an output. The time from capturing an input to generating an output is the system loop time. Rule-evaluation time is bounded by the system loop time. This in turn, is limited by the Sampling Theorem, which requires sampling frequency to be at least twice the highest frequency of the sampled signal. The highest frequencies of the input and the output signals define the

*Continued on page 54*

*Embedded Systems: Part 4*

# Rules are clear for fuzzy controllers

*Continued from page 52*
limit of execution time.

For example, if the input is 1 kHz, the sampling frequency must be a minimum of 2 kHz. That means the microcontroller must "go through the loop" every 1/(2 kHz), or 500 μs. Not all of the 500 μs can be allotted to the inferencing process.

**Performance determinants**

During this time, the microcontroller must capture the input data, perform the inferencing and generate the output. The performance requirements are bounded by the number of rules to be evaluated and the sampling period on the input and output. Elements that affect this performance are fast and autonomous peripherals, adequate internal data storage and efficient architecture for logical operations. Less time spent on servicing input and output provides more time to inference.

CPU execution gates the performance. One architectural feature that improves performance is a register file, which permits storing the term points and the membership of terms in the registers. As a result, performance is enhanced by eliminating external memory cycles.

Usually, no more than nine terms are used, and each has four points per term. Therefore, the amount of internal RAM required is 36 bytes or less per term. Seven is a more typical number of terms, requiring only 28 bytes. A four-input system would then require 112 bytes to map all terms.

**Defuzzifying the output**

Defuzzification is performed by calculating the area under curves of the output terms. Multiply and accumulate functions are used to calculate the crisp output value, and 16-bit mathematics will enhance performance. For large rule sets or minimum loop time, 16-bit microcontrollers should be considered.

Software implements fuzzy logic with standard microcontroller instructions. The CPU does the fuzzification, inference and defuzzification via software while the peripherals capture the input data and provide an output. A single-chip microcontroller such as the Intel MCS-96 can perform both functions—interface and control—using software. The advantages of using a standard microcontroller are cost, availability, a well-known standard architecture and programming tools that are popular and easy to use.